Experimental Study of Artificial Neural Networks Using a Digital Memristor Simulator

Vasileios Ntinas, Ioannis Vourkas, *Member, IEEE*, Angel Abusleme, *Member, IEEE*, Georgios Ch. Sirakoulis, *Member, IEEE*, and Antonio Rubio, *Senior Member, IEEE*

***Abstract*—This paper presents a fully digital implementation of a memristor hardware simulator, as the core of an emulator, based on a behavioral model of voltage-controlled threshold-type bipolar memristors. Compared to other analog solutions, the proposed digital design is compact, easily reconfigurable, demonstrates very good matching with the mathematical model on which it is based, and complies with all the required features for memristor emulators. We validated its functionality using Altera Quartus II and ModelSim tools targeting low-cost yet powerful field programmable gate array (FPGA) families. We tested its suitability for complex memristive circuits as well as its synapse functioning in artificial neural networks (ANNs), implementing examples of associative memory and unsupervised learning of spatio-temporal correlations in parallel input streams using a simplified STDP. We provide the full circuit schematics of all our digital circuit designs and comment on the required hardware resources and their scaling trends, thus presenting a design framework for applications based on our hardware simulator.**

***Index Terms*—associative memory, computing, emulator, memristor, neural network, neuromorphic, resistive switching**

I. Introduction

THE existence of the 4th fundamental circuit element was postulated by Chua in 1971 and was termed "memristor" (short for "memory resistor") [1]. Today, the term "memristor" usually refers to any resistance-switching (RS) device that complies with a particular set of requirements known as memristor "fingerprints" [2], regardless of the fabrication details [3]. The hysteretic RS properties of oxides sandwiched between metal electrodes was well known though even from the 60s, as seen in relevant publications by Hickmott and Argall [4], [5]. Chua's theory of the memristor was however connected with experimental devices only in 2008 by Hewlett-Packard Laboratories and their work on TiO_2 RS devices [6]. Memristors are now considered a rapidly emerging technology [7] that creates opportunities to realize innovative circuits and systems with applications such as nonvolatile memory [8], [9], adaptive circuits [10], [11], signal processing [12], and logic/computing [13-15].

The memristor has also been proposed as the electronic analog of biological synapses [16], [17]. It is essentially a resistor with memory; it is nonvolatile (although volatile devices have been also reported [18] and the specific theory can be found in [19]), its response depends on its whole dynamical history, and it demonstrates a continuous set of resistance values, making it ideal for tuning synaptic weights of artificial neural networks (ANNs) [20-22]. An ANN is a data-processing model based on the biological nervous system, implemented as a parallel and distributed network of simple nonlinear processing units [23]. Hardware (HW) implementation of ANNs is an important step toward obtaining human-brain-like functionalities at circuit level. The basic components of ANNs are neurons and synapses, whose circuit realization should mainly be compact to make scaling up to approach the total biological device numbers (~10^{11} neurons and ~10^{15} synapses in human brain) feasible.

As typically happens with all new electronic devices, modeling and simulation are the first steps to exploring memristors' general attributes, verifying theoretical aspects, and understanding the effect of different model parameter values. In this context, several either behavioral or physics-based (usually SPICE-compatible) models have been developed [24-27]. Lab experiments with fabricated memristors are the next step. However, memristor technology is still in progress and device fabrication implies considerable costs and difficulties. Furthermore, the memristors commercially available to date are quite expensive and still not very reliable [28]. Consequently, research and development have largely focused on various (mostly analog) HW emulators [29-31] that facilitate the experimental exploration of memristive behavior.

According to [30], the required features for a memristor emulator are: i) a wide memristance range; ii) nonvolatility; iii) initial state configurability; iv) floating operation; v) operability for high-frequency and continuous input signals;

and vi) support interconnection with other components. Taking these six electrical requirements into account, in this paper we build upon our previous work in [32] and develop a fully digital memristor hardware simulator based on a behavioral model of voltage-controlled threshold-type bipolar memristors [27]. The presented electronic module constitutes the core of a digital memristor emulator, which further requires interface circuitry to permit connection to external circuits as a two terminal element, such as in [27], [31]. We conducted all the required verification tests and validated its functionality using Altera Quartus II and ModelSim software (SW), targeting low-cost yet powerful field programmable gate array (FPGA) families. The FPGAs are reconfigurable electronic platforms well-suited to implement ANNs [33], [34] owing to their HW flexibility, which allows rapid prototyping of different ANN topologies and implementation strategies. In this context, we chose the FPGA as the target electronic platform and showed that our design is suitable for FPGA-based ANNs. Our motivation was to design and implement digital HW electronic synapses particularly based on memristive dynamics and prove their suitability and applicability to a variety of ANN-based applications. Compared to other emulation approaches from the recent literature, this digital design is compact, easily reconfigurable, demonstrates excellent matching with the memristor model on which it is based [27], and complies with all the aforementioned electrical requirements (i to vi). Moreover, we tested its suitability for anti-serial memristive interconnections [35] and proved its synapse functioning in single-layer perceptron, implementing examples of associative memory and a simplified variation of spike timing dependent plasticity (STDP) [36] unsupervised learning of spatio-temporal correlations in parallel input streams, following previous demonstrations in [31] and [37], respectively. We present the schematics of our digital circuit designs and comment on the required HW resource scaling, thereby providing a complete design framework for such memristor emulator-based ANNs.

II. Memristor Digital Hardware Simulator

A. The Behavioral Memristor Model

Even though any mathematical model [25] could serve as a basis of our digital implementation, the developed digital simulator is based on (and meets all the characteristics of) the behavioral threshold-type bipolar memristor model proposed in [27], described by the following equations:

$$i(t) = R^{-1} \cdot v(t) \tag{1}$$

$$\dot{R} = b \cdot v + \frac{1}{2}(a-b) \cdot (|v+v_T| - |v-v_T|) \cdot \theta(R - R_{ON}) \cdot \theta(R_{OFF} - R) \tag{2}$$

It is a behavioral model of a voltage-controlled time-invariant memristor whose memristance change rate is given by the piece-wise linear equation (2). Equation (1) reflects the *state-dependent Ohm's Law*, where $i(t)$ is the flowing current and $v(t)$ is the voltage drop on the memristor, whereas R is the memristance and, at the same time, the system's only state variable. Equation (2) depends only on $v(t)$ and R. As shown in Fig. 1, its value changes at different rates when the applied voltage is either higher or lower than the threshold voltage v_T, limited by upper and lower boundaries, namely R_{ON} and R_{OFF} (i.e., R_{MIN} and R_{MAX}). The latter is accomplished with the use of the step function $\theta()$ in (2), which indicates that R can change only between its limiting values. The resistance change rate of threshold-type switching memristors is very fast above (and negligibly slow below) the threshold v_T, which here, for the purposes of simplicity, is considered symmetric for both the SET ($R_{OFF} \rightarrow R_{ON}$) and RESET ($R_{ON} \rightarrow R_{OFF}$) transitions. The constants a and b in (2) define this change rate when $|v(t)| < v_T$ or $|v(t)| > v_T$, respectively, with a, $b < 0$ and $|a| < |b|$. Thus, the resistance decreases when the memristor is forward-biased and increases when it is reverse-biased. Hereinafter we will refer to a memristor being forward/reverse-biased when the voltage at the top/bottom terminal is higher than that on its bottom/top terminal; the bottom terminal is denoted by the thick black line in the circuit schematic (see Fig. 2).

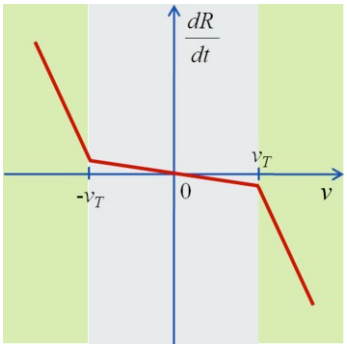


Fig. 1. Qualitative graph for the memristance change rate in (2) as a function of the applied voltage.

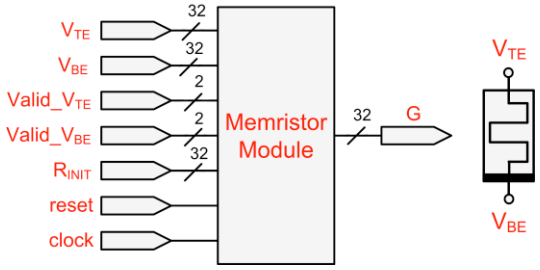


Fig. 2. Compact memristor hardware simulator block diagram.

B. Circuit Implementation

Fig. 2 shows the compact block diagram of the memristor module. The input signals include: the top and bottom electrode voltage (V_{TE} and V_{BE}), these being the basic two whether the applied voltage is valid or has been left floating (the need for two bits instead of one is explained in Section

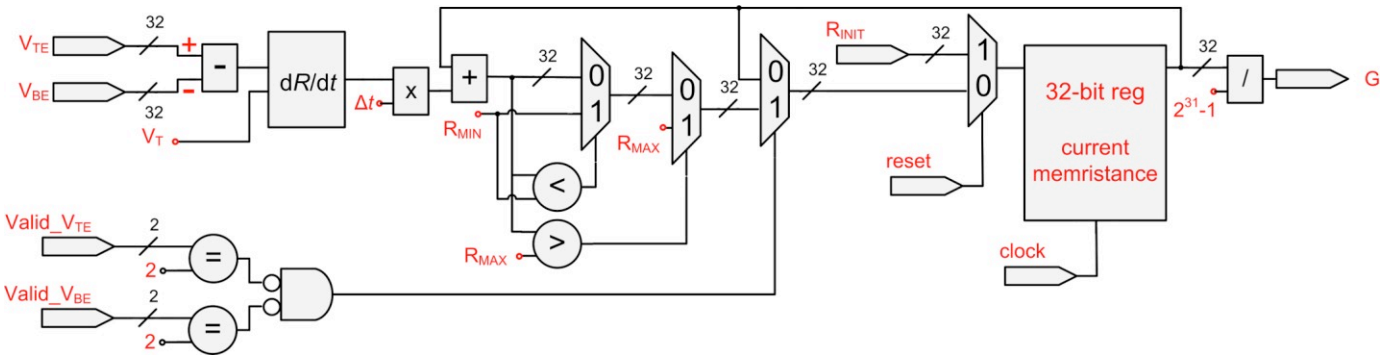


Fig. 3. Detailed memristor hardware simulator block diagram.

TABLE I
FPGA IMPLEMENTATION LOGIC DATA

Family / Device	Cyclone II / EP2C70F672C6
Total Logic Elements	3266 (5%)
Total Registers	32 (<1%)

IV). Unlike in [32], in this version of the simulator design, the output signal in most cases concerns the memductance $G = R^{-1}$ because using G instead of R enables a simplified circuit design, lower HW resource requirements, and greater computational precision, as shown in the following sections. The model-specific parameters α, b, v_T, R_{MIN}, and R_{MAX} are defined as internal constants (stored in memory) since we assume they are device/module-specific. Preferring power-of-2 values for such internal constants and for all the auxiliary variables used in multiplications/divisions significantly minimizes the required HW resources.

Fig. 3 shows a more detailed implementation block diagram. The input voltage is used to compute the derivative of R as in (2) and multiply it by a properly selected integration time step Δt, which also defines the maximum supported input signal frequency. The result is added to the current R value and, once the out-of-bound and floating terminal controls have been performed (the number $(2)_{10} = (10)_2$ shown at the bottom left of Fig. 3 corresponds to the case of a floating connection), is stored in the corresponding register. The result of such controls are selection bits in multiplexers, i.e. blocks that conditionally pass always one of the inputs to the output according to the selection bit. For data representation, we use up to 32-bit integers and thus guarantee a wide value range and adequate precision for all important parameters (e.g., a wide memristance range), while preventing under/overflow during computations through the appropriate selection of α, b, and Δt. In integer computations, the fractional part of any result is truncated. Therefore, we represent resistance values in mΩ instead of Ω (i.e., 5000 instead of 5), and voltages in μV instead of V, to create the necessary precision. Moreover, the use of an auxiliary variable (the max integer) is shown in the last block in Fig. 3. In fact, this is a necessary transformation to obtain a valid G value since inverting R in digital HW would simply give zero as a result. Therefore, in order to keep this information from the inversion, we shift the result via a

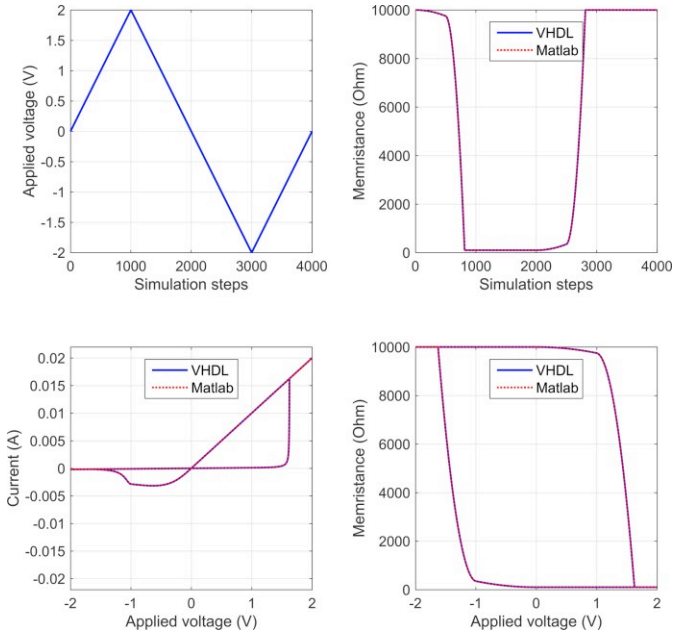
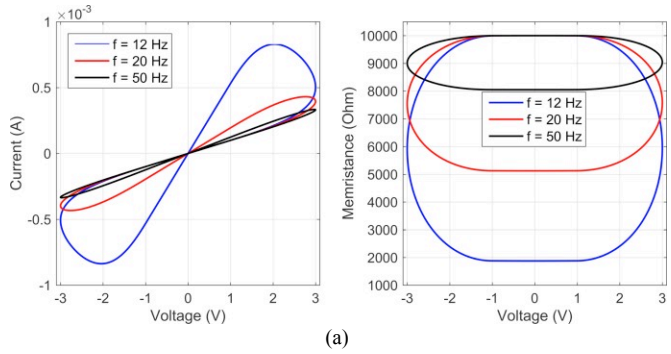


Fig. 4. Comparison between the mathematical model (Matlab) and the hardware simulator response (VHDL). α = -2000 Ω/(V×s), b = -190000 Ω/(V×s), Δt = 0.0005s, v_T = 1V, R_{MIN} = 100Ω and R_{MAX} = 10KΩ.

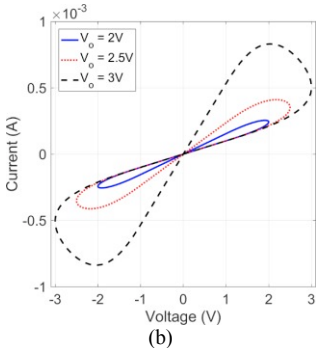
multiplication with a very large number defined as power of two (i.e. if $R = 10^4\Omega$, then it is 10^7mΩ and we compute $G = (2^{31}\text{-}1)/10^7) = 214$S). The result is of course not the correct G as the latter includes the shifting operation. However, this is not really a problem and the practical meaning of this transformation is further explained in Section IV. Generally the use of such auxiliary variables is omitted in the block diagrams for the sake of simplicity. Basic information about the HW resources of the memristor module for a specific FPGA, is given in Table I (% refers to the percentage of the total available resources in the FPGA that are being used).

III. BEHAVIORAL VERIFICATION TESTS

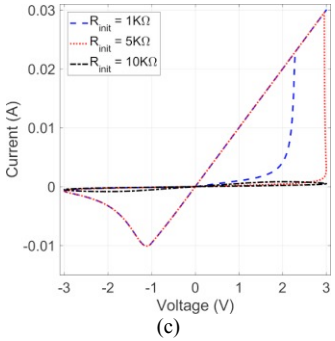
This section presents the series of functional verification tests carried out to prove the HW simulator's proper functioning and matching with the mathematical model, its compliance with the characteristic fingerprints, the multi-level tuning property required for analog applications, and its suitability for complex interconnections. All the measurements



(a)



(b)



(c)

Fig. 5. (a) *i-v* and *R-v* plot concerning the response of the HW simulator for various frequencies of the input voltage $v(t) = 3sin(2\pi ft)$ when $R_{INIT} = R_{MAX}$. (b) *i-v* plot for varying input amplitudes V_o of $v(t)$ when $f = 12$ Hz and $R_{INIT} = R_{MAX}$. (c) *i-v* plot for varying initial conditions (R_{INIT}) when $V_o = 3$V and $f = 12$ Hz. In all scenarios we used $\alpha = -2000$ Ω/(V×s), $b = -190000$ Ω/(V×s), $\Delta t = 0.0001$s, $v_T = 1$V, $R_{MIN} = 100\Omega$ and $R_{MAX} = 10$KΩ.

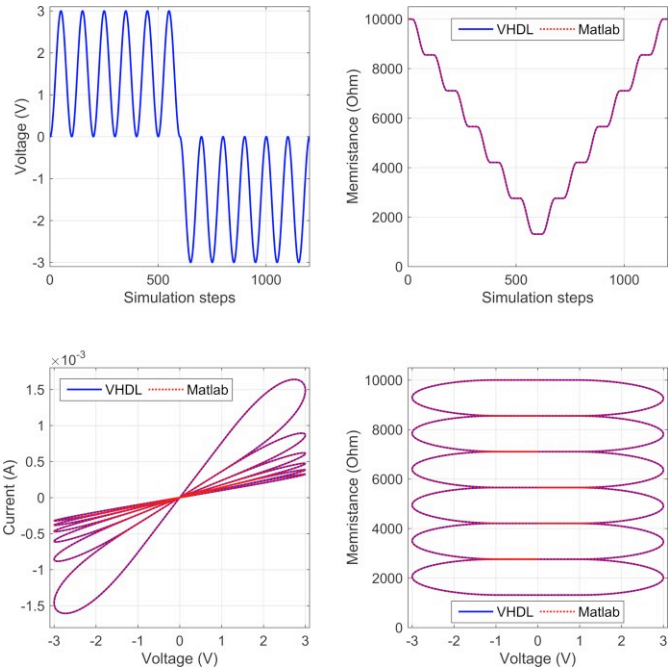


Fig. 6. Multi-state programming feature. Comparison between the mathematical base model (Matlab) and the HW simulator response (VHDL). $\alpha = -2000$ Ω/(V×s), $b = -190000$ Ω/(V×s), $\Delta t = 0.0001$s, $v_T = 1$V, $R_{MIN} = 100\Omega$ and R_{MAX}=10KΩ, whereas the applied voltage was $v(t) = \pm 3\sin^2(100\pi t)$.

for our design use the ModelSim HW simulation data for a target FPGA device and 100MHz clock frequency. These data were collected via proper Matlab scripts and compared with the reference model data. The ModelSim input files were prepared using the Matlab HDL coder. In all demonstrated measurements, the reset phase when the system was initialized was simply omitted.

A. Match with the Mathematical Memristor Model

It is important to guarantee an exact match between the HW simulator's response and data from the memristor model on which it is based, regardless of the characteristics of the input voltage. Fig. 4 shows a relevant comparison concerning a triangular input voltage pulse. The HW simulator's response matches very well that of the model. The available precision during computations with integer variables in our design guarantees infinitesimal error. Such precise matching is obtained for different input pulse types and frequencies, provided that the selected model parameter values do not cause under/overflow problems.

B. Compliance with the Characteristic Fingerprints

We proved that our HW simulator indeed behaves as a memristor by testing its compliance with memristor fingerprints [3]. Fig. 5(a) shows a set of current-voltage curves, along with the memristance-voltage curves, for different frequencies of the input sinusoidal voltage. All curves in the *i-v* plane are pinched at the origin $(i, v) = (0, 0)$, i.e., there is no phase shift between the $i(t)$ and $v(t)$

waveforms. This is valid for all amplitudes and frequencies of the input signal (see Fig. 5(b)), and for any possible initial condition of memristor (see Fig. 5(c)), as seen in the *R*-*v* and *i*- *v* planes. Moreover, the so-called *single-valued function limiting phenomenon* is confirmed with the collapsing hysteresis loops in the *i*-*v* graph of Fig. 5(a). As the sweep frequency *f* increases, the area of each lobe of the pinched hysteresis loop shrinks, such that the memristance function degenerates to a straight line (tends to a single value) as *f* increases towards infinity.

C. Multi-State Tuning Capability

One of the main reasons why memristors have been proposed as the electronic analog of biological synapses is because they demonstrate a continuous set of resistance values and are thus ideal for representing synaptic weights. Such multi-level tuning capability is crucial for analog applications. Therefore, we tested the multi-state tuning capability of our design, which is required to model synapse functioning. The relevant HW simulation results are shown in Fig. 6. Multiple continuous states are obtained via successive short voltage pulses of the same polarity. In our case, we show seven distinct memristance levels achieved with a $\pm V_o \sin^2(100\pi t)$ pulse train (V_o = 3V). Since higher applied voltages cause faster switching, decreasing the pulse amplitude while ensuring it remains above the threshold results in much closer distinct memristance levels. Shortening the pulse duration while maintaining the same amplitude has a similar effect. As shown in Fig. 6, in all such cases the module's multi-level switching response matches very well the reference model.

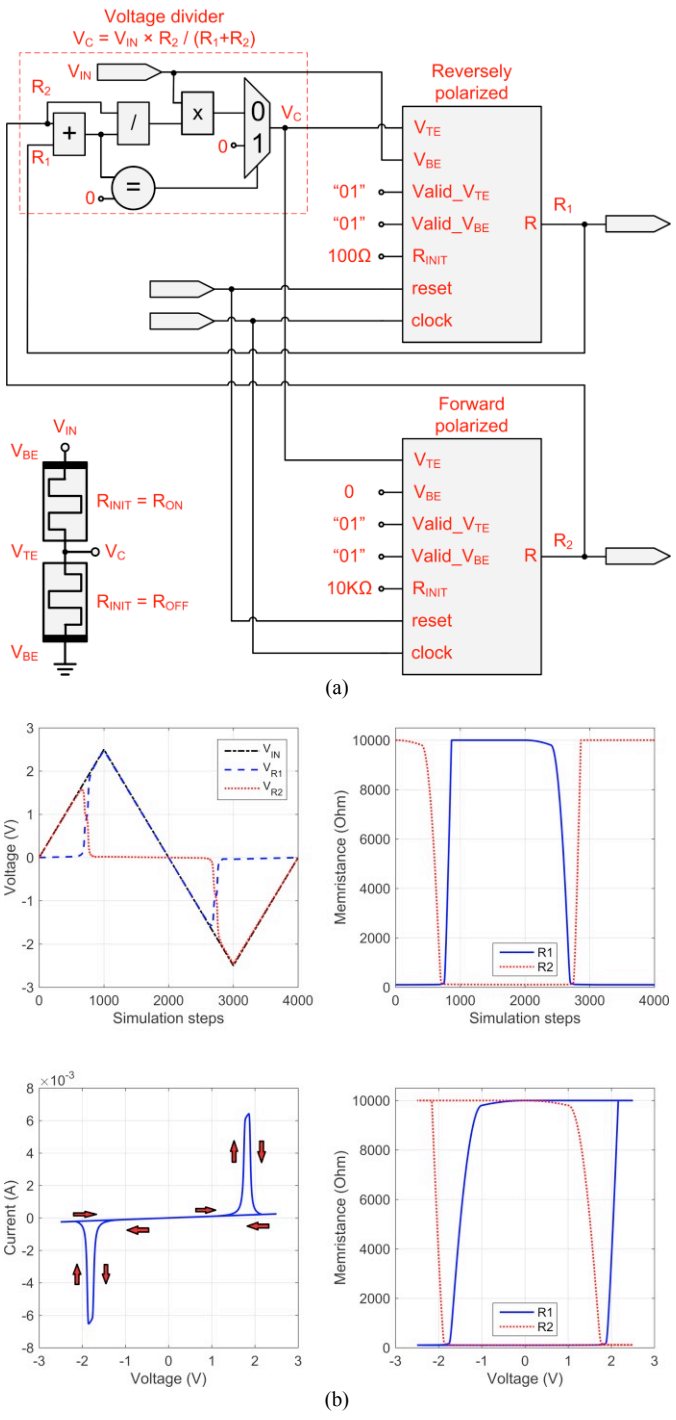


Fig. 7. Anti-serial connected memristors. (a) Block diagram showing the interconnection of the electronic modules and the additional components for the computation of the voltage divider equation. The inset shows the equivalent circuit using the memristor symbol. (b) Simulation results showing the applied voltage V_{IN} and the voltage drop on the two memristors, the composite *i-v* plot, and the HW simulators' memristance evolution with time and with the input voltage, respectively. α = -2000 Ω/(V×s), b = -190000 Ω/(V×s), Δt = 0.0005s, v_T = 1V, R_{MIN} = 100Ω and R_{MAX} = 10KΩ.

D. Complex Device Interconnection

In keeping with the electrical requirements for memristor emulators mentioned in the Introduction, this verification test checked the possibility of interconnection with other components, i.e., whether the developed HW simulator can be connected to other devices or emulators, which is essential for it to be used in more sophisticated circuit configurations. It is worth mentioning that unless an interface circuit is added, such as for example an ADC and a digital potentiometer, similar to that shown by Pershin and Di Ventra in [27] and [31], then our implementation cannot be electrically connected to an external circuit as a complete digital memristor emulator.

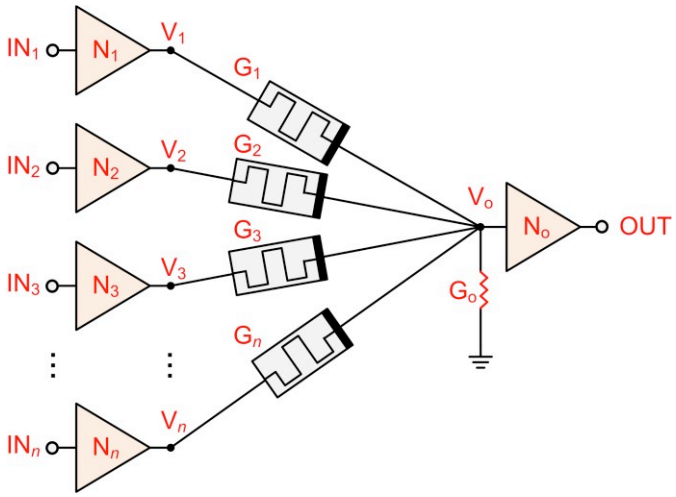


Fig. 8. Generic implementation scheme for a single-layer ANN with n input neurons N_i (i = 1,…, n) connected to an output neuron N_o via n memristors. G_o is a small conductance used in the Kirchhoff's Current Law computation.

To this end, we studied the suitability of the simulator for the complementary resistive switch (CRS) configuration [35]. A CRS consists of two memristors connected in series but with opposite polarities (anti-serially), hereinafter called the forward-polarized memristor (FPM) and the reverse-polarized memristor (RPM). The CRS is a comprehensive enough test, also easy to implement, which allows to confirm both the interconnection property and the polarity-dependent switching of multiple such modules combined together. Memristors with opposite polarities demonstrate reverse behavior to the applied signal; i.e., during one period of the AC input voltage, complementary devices reciprocally change their states.

Fig. 7(a) shows the block diagram of the CRS configuration. For simplicity, we have defined the output of the simulators as the memristance R instead of the memductance G shown in Fig. 3. Apart from the two memristor modules, the system requires a combinational part to calculate the voltage V_C on the common intermediate node of the memristors, i.e., to compute the voltage divider equation. The latter receives V_{IN} as input and uses the current state of the two memristors (R_1 and R_2) to drive them with the corresponding V_{TE} value. The sum of R_1 and R_2 is computed first and then the fraction, which is multiplied by the input voltage, as shown in the inset. A MUX is used to prevent invalid results by division with zero. The flag inputs "01" denote there are no floating electrodes. The two modules are set to the FPM/RPM = R_{OFF}/R_{ON} state during initialization. The inset shows the equivalent circuit schematic for such connection, using the memristor symbol for clarity.

Fig. 7(b) shows the HW simulation results. The positive

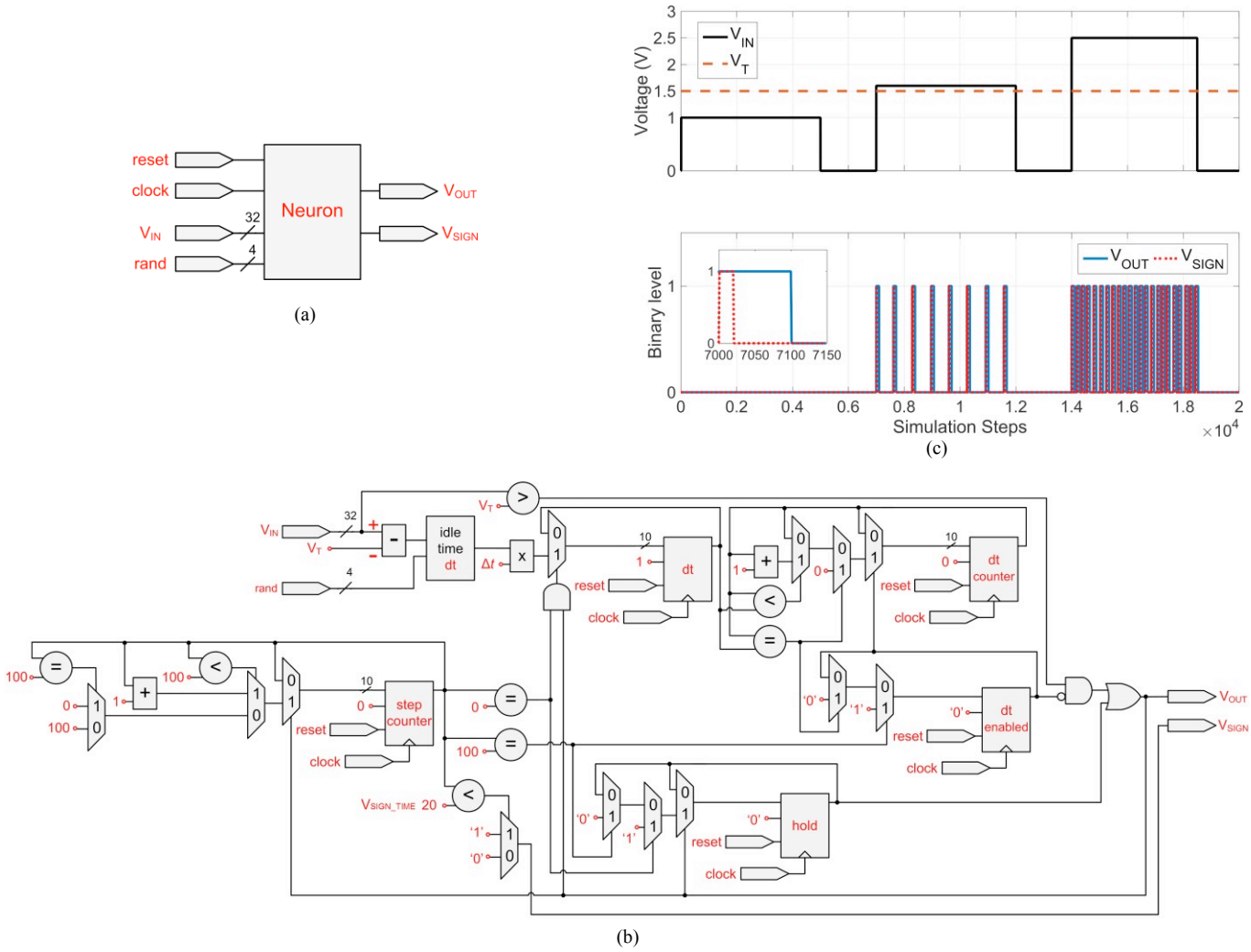


Fig. 9. (a) Compact and (b) detailed block diagram of the neuron module based on the model described in [31]; (c) HW simulation results for the neuron's behavior for different amplitudes of the input voltage V_{IN}. For the purposes of clarity, the inset in the output plot focuses on a specific excitation cycle.

part of the triangular input V_{IN} creates the necessary conditions first to change the state of the FPM (R_2) from R_{OFF} to R_{ON} and, later, that of the RPM (R_1) from R_{ON} to R_{OFF}, resulting in a flipped resistive configuration. The memristors then exhibit an ohmic behavior until the applied voltage exceeds the respective negative thresholds and forces them to successively switch to their initial states. In Fig. 7(b) we also show the perfectly symmetric composite *i-v* curve. Overall, the results confirm the reproduction of the CRS operation.

IV. Memristive Artificial Neural Networks

A. Circuit Implementation

In this section we use the developed hardware simulator as a synapse and present an implementation scheme for the perceptron topology shown in Fig. 8. All the additional electronic modules, like the previously presented for memristor, concern digital designs easily tuned and generically built to facilitate the development of single- or multi-layer ANNs on FPGA devices for several applications.

Specifically in the single-layer ANN example shown in Fig. 8, the input neurons N_i (i = 1, …, n) are connected with an output neuron N_o via synapses (memristors), while the output signal OUT is determined by the applied input signals IN_i and the strength of the synaptic connections G_i, which weigh the potentials V_i. Biological neurons generally have receptor (synaptic) and action potentials. When the receptor potential at the input of an idle (not firing) neuron exceeds a given threshold, the neuron is excited and starts firing, i.e. starts emitting forward and backward fixed-amplitude action pulses. As we will show in the following ANN examples, the back-propagating pulses are responsible for synaptic tuning and, therefore, for continuous re-learning. For the purposes of our experiments, we developed a digital electronic version of a neuron model as described next and in more detail in [31].

Figs. 9(a) and 9(b) show the compact and detailed block diagrams of the developed neuron module, which receives two inputs. V_{IN} is the receptor potential, which is constantly monitored; once it exceeds a threshold value V_T (defined as an internal constant, i.e., stored in memory), forward and backward fixed-amplitude action pulses are generated. However, the computed pulse separation (the waiting/idle time) varies according to the strength of the receptor input stimulus and a random parameter, which is the module's

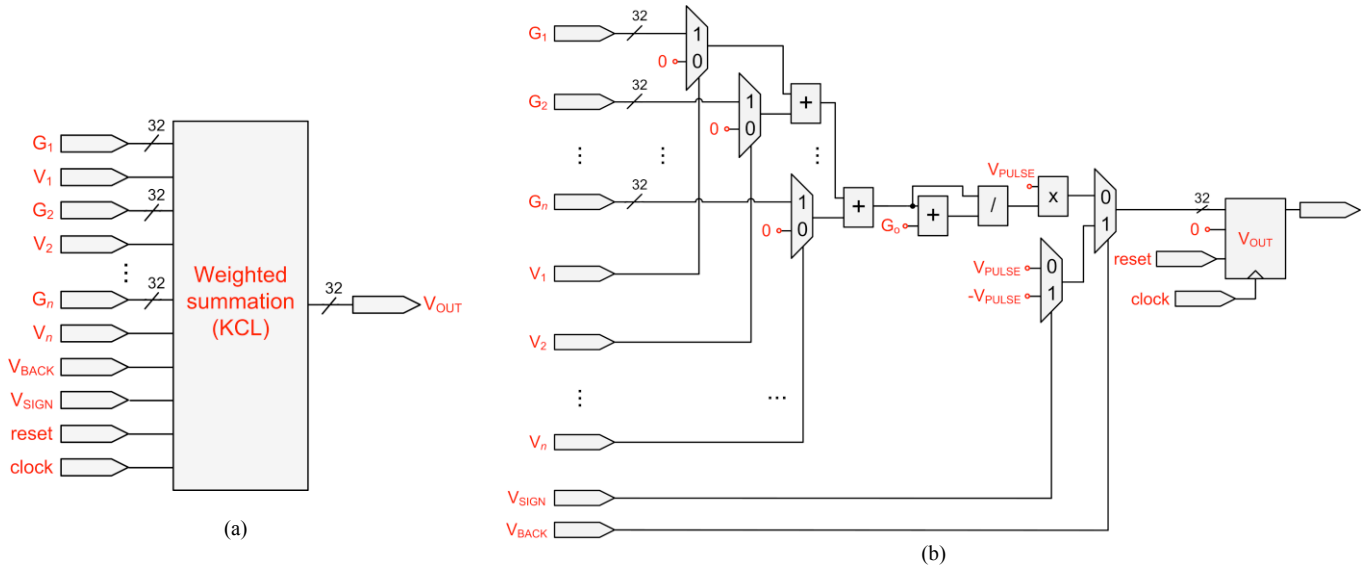


Fig. 10. (a) Compact and (b) detailed block diagram of the Kirchhoff's Current Law (KCL) computation (weighted summation) module giving the voltage at node V_o of Fig. 8. We set $G_o = (2^{31}-1)/R_o$, and assumed a large resistance $R_o \approx 1M\Omega$.

second input. The output V_{OUT} of the neuron is only indicating whether the neuron is excited or idle. There are three possible cases for the memristor terminals to be considered since, according to [31], when a neuron is idle (not firing), its output terminal (e.g., V_i for input neuron N_i in Fig. 8) becomes floating. However, while still in idle state, its input terminal (i.e. the common node of all the synapses, V_o for output neuron N_o in Fig. 8) may still be connected and receiving input stimuli caused by the pre-synaptic pulses in case one of the input neurons is firing. Therefore, defining V_o simply floating due to the neuron being idle, is incorrect. We overcame this requirement via local bit-processing in-between the V_{OUT} of a neuron module and the Valid_$V_{TE/BE}$ flag of a memristor module, thereby driving two bits to the flag inputs, enough to model the three possible cases (connected, floating, and connected while neuron is idle, as explained in Table II). This enabled us to keep the implementation of neurons general and their complexity low.

Moreover, the fixed amplitude of the action pulses is not defined inside the neurons but rather is set externally and thus applied directly to the corresponding terminal of every memristor, as shown below. Unlike in our previous work [32], here we included a second neuron output V_{SIGN} indicating whether the back-propagating pulses have a negative sign. As we will show below in our ANN examples, unlike in [31], [32], this important added property makes it possible both to increase and decrease the synaptic weights through the simplified STDP scheme implemented here, previously proposed in [38] (although action potentials resembling more the true spike waveforms found in biological neural systems, as presented in [36], [39], [40], could be implemented as well by more HW resources). When both input and output neurons are firing, the resulting voltage drop (e.g., V_i - V_o in Fig. 8) on the memristors is of constant amplitude but different durations, depending on the timing of the voltage signals at the two sides of the synapses. On the other hand, when the output

TABLE II
POSSIBLE SITUATIONS FOR OUTPUT NEURON'S TERMINALS

Neuron's state	Input terminal (V_o)	Output terminal (OUT)
Idle	Connected (V_{OUT} of KCL module)	Floating
Excited	Connected ($\pm V_{PULSE}$)	Connected (V_{PULSE})

neuron is not firing, then V_o varies depending on the state of input neurons and their synapses.

More specifically, according to Fig. 9(b), the neuron operation is determined based on two counters defining the total excitation time steps and the pulse separation time, mentioned before. We arbitrarily set the *excitation time* to 100 clock steps. This duration is defined by the *step counter* (bottom left in the figure), which starts counting when the neuron becomes excited. Inside the neuron, the number of steps when the V_{SIGN} is '1' is also defined as V_{SIGN_TIME} which is assumed to be 20 steps in Fig. 9(b), a value chosen based on trial and error to improve the results obtained in the application examples shown next; for the rest of the excitation cycle steps, V_{SIGN} is '0'. The *hold* register is responsible for keeping the neuron excited (i.e. neuron's output V_{OUT} = '1') for 100 steps. Moreover, when the *step counter* starts, the *dt* register stores the idle/waiting time, which is computed according to a formula proposed in [31] and converted to true simulation time via multiplication with the integration time step Δt. When the *step counter* reaches 100, the *hold* register becomes '0' but the *dt enabled* becomes '1', thereby activating the *dt counter*, which is responsible for keeping the neuron idle for a waiting time (refractory period) equal to *dt*. When *dt counter* = *dt*, then the *dt enabled* becomes '0'. A positive difference V_{IN}-V_T can activate the neuron provided that the *dt enabled* register is '0', i.e., that the neuron is not in refractory period. Adjusting the V_{SIGN_TIME} during module instantiation makes it possible to create the desired time ratio

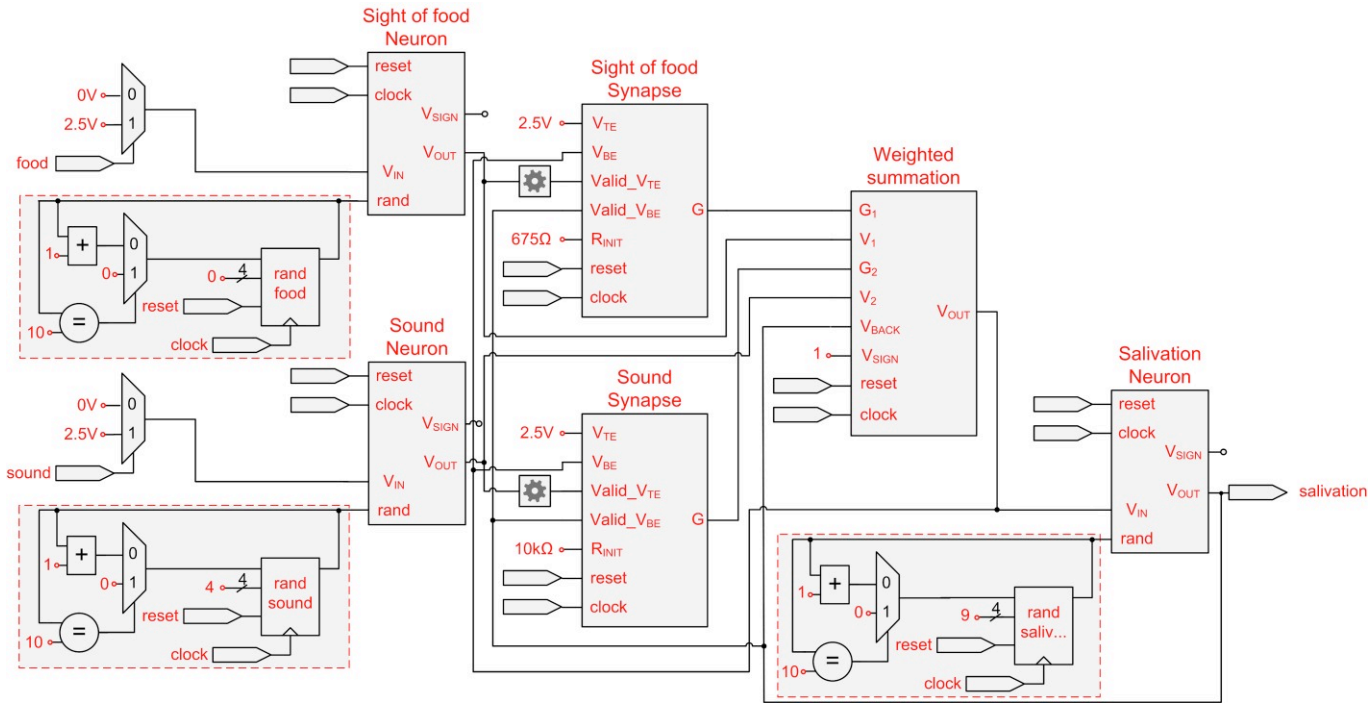


Fig. 11. Block-level circuit topology implementing the linear perceptron with three neurons and two synapses, according to [31], using the neuron, the memristor, and the weighted summation (KCL) electronic modules. The small gear driving the memristor Valid_V_{TE} inputs denotes local bit-processing mentioned in text.

of negative and positive back-propagating action pulses during excitation, as proposed in [38]. HW simulation results of the neuron's behavior are shown in Fig. 9(c). We present the action potentials caused by different amplitudes of the input V_{IN}, whereas for the *rand* input we use a series of randomly generated integers ∈ [0, 10]. When $V_{IN}<V_T$ there is no firing. Otherwise, the average pulse separation decreases as the input amplitude increases. The effect of the *rand* input on the *dt* computation is more evident at higher firing rates. Fig. 9(c) also shows the V_{SIGN} output, which, in this simulation scenario, is '1' for the first 20 of the 100 excitation time steps.

In short, looking back at Fig. 8, we can conclude that when the input neurons are firing, the memristors receive positive action potentials V_i at their TE, whereas when they are idle, these terminals are assumed to be floating. On the other hand, when the output neuron is firing, the memristors receive either positive or negative action potentials V_o at their BE, but when the neuron is idle, the receptor potential V_o at its input terminal needs to be calculated (see Table II). We saw a similar problem before in the CRS example. Whenever there is a shared node among many interconnected memristors, an additional module responsible for computing the instant potential at that node is required. Therefore, for the purposes of our experiments, we developed an electronic version of the Kirchhoff's Current Law (KCL) computation (weighted summation) shown in Fig. 10. As can be seen in Fig. 10(a), this module receives the current memductance of every memristor G_i (which includes the shifted operation as explained previously) and a series of bits V_i, which are the outputs of the input neurons (V_{OUT}). The output (i.e., the V_o potential in Fig. 8) is updated using the KCL equation when the input V_{BACK} is '0', i.e., when the output neuron is not excited and there is thus no back-propagating pulses. Otherwise, i.e. when the output neuron is firing, the output becomes equal to the predefined amplitude $\pm V_{PULSE}$ depending on the V_{SIGN} input. V_{PULSE} is the fixed amplitude assumed for both the forward and backward action pulses in our ANNs. As shown in Fig. 10(b), we built this module in a generic manner in order to receive an arbitrary number of inputs, provided that the internal KCL computations do not cause overflow. According to the KCL formula $V_o = (V_1G_1 + V_2G_2 + \ldots + V_nG_n) / (G_o + G_1 + G_2 + \ldots + G_n)$, first we sequentially compute the conditional sum of all G_i (depending on the corresponding V_i which serves as MUX selection bit) and G_o. Then we compute the fraction and eventually multiply it by the fixed amplitude V_{PULSE}. Since this formula has G both in the numerator and denominator, the previous shifting transformation is inherently removed and does not affect the result of this computation. The reason we prefer G instead of R is to simplify the implementation of the KCL block as several more divisions are required if R is used in this formula, while we also noted even a better precision. In the next section, we use all these modules and present two ANN example configurations for two different applications.

B. ANN Example for Associative Memory

Following [31], which demonstrates a neural network implementing the famous "Pavlov's dog" experiment [41], here we present a similar ANN implementation as a proof of concept of associative memory. Fig. 11 presents the block-level circuit topology implementing the linear perceptron with two input neurons and one output neuron connected via two synapses. A few details about the Pavlov's dog target experiment [31], [41]: Initially, a dog salivates only at the

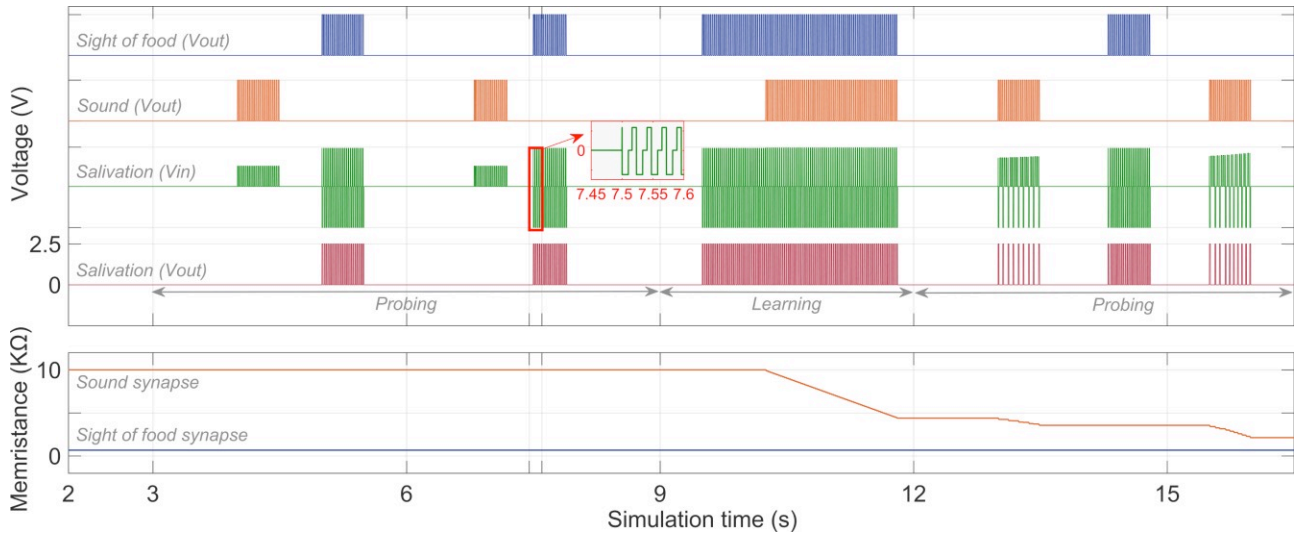


Fig. 12. HW simulation results for the circuit in Fig. 11. Parameter values of the memristor model (α = 0 Ω/(V×s), b = -15000 Ω/(V×s), Δt = 0.0001s, v_T = 4V, R_{MIN} = 675Ω and R_{MAX} = 10KΩ.), the voltage amplitudes, and the time duration of the experiment were set following the experiment in [31], for the purposes of comparison. The graphs of the top plot show the output voltage V_{OUT} of the three neurons and the input voltage V_{IN} of the salivation neuron (combination of a positive amplitude receptor potential and of a negative amplitude back propagating action potential). The top three curves in the plot were displaced vertically for clarity. The bottom plot shows the synapse memristance evolution with time.

sight of food. However, if the sight of food is accompanied by a particular sound for a certain period of time, then the dog learns to associate the sound with food; hence, thereafter salivation can be triggered using only the sound (Hebbian learning rule). The values for the amplitude of the action pulses (2.5V), the internal threshold of the neurons (1.5V), and the initial synapse memristance are all set as in [31] for the purposes of comparison. The amplitude of the action potentials is set directly at the V_{TE} terminal of every memristor emulator. The *food* and *sound* 2.5V input pulse trains define the receptor potential of the input neurons, whose output V_{OUT} is driven to the TE flag input of the corresponding memristor modules (after undergoing local bit-processing), as well as to the weighted summation (KCL) module. The latter also receives the current memductance G of the memristors. Thus, the memristor V_{BE} voltage is in fact the output of the KCL module, which is also driven to the input terminal of the *salivation* neuron. In order to properly consider the cases shown in Table II and correctly define whether the memristor terminals are floating or not, Table III explains the small *gear* box bit processing operation, shown in Fig. 11. Practically, b_0 is the output of the neuron and b_1 is a bit we append defining whether this is an output neuron (b_1 = 0) or input neuron (b_1=1). Hence, inside the memristor module, only the combination "10" denotes a floating TE.

In this ANN, the V_{SIGN} output of the neurons is not used because only negative backward action pulses are required, so the V_{SIGN} input of the KCL module is constantly set to '1'. The output V_{OUT} of the *salivation* neuron is driven to the V_{BACK} input of the KCL module. Regarding the *rand* input of the neurons, in [31], this input was driven by a random value created within a microcontroller. However, in the absence of any particular pseudo-random number generation module here and in order to avoid the prior preparation of random values at software level, we instead used a 0-10 counter (the circuit

TABLE III
2-BIT ENCODING OF VALID TE/BE FLAGS

Memristor electrode	Connected neuron's state	Representation as "b_1b_0"
Valid_V_{TE}	Idle	"10" (Floating)
	Excited	"11" (Not Floating)
Valid_V_{BE}	Idle	"00" (Not Floating)
	Excited	"01" (Not Floating)

within a dashed line box in Fig. 11), which we purposely initialized at a different value for every neuron (i.e., at 0, 4, and 9, respectively). Hence, we created a different sequence of *rand* input values for each neuron, which still affected the neuron waiting time *dt* satisfactorily for the purposes of our experiments.

Fig. 12 shows the HW simulation results, which are very much in line with the results published in [31] and confirm the development of associative memory. During the first probing phase, it is evident that salivation is uniquely caused by the *sight of food* neuron. This happens due to the initial high conductance of the connecting synapse, which permits the pulse at the input of the *salivation* neuron to exceed the internal activation threshold. There is no synapse state change in this phase as it can be observed in the plot at the bottom showing the memristance evolution with time. This happens because the *sight of food* synapse is already in R_{ON}, whereas the *sound* neuron is idle while the *salivation* neuron is excited, meaning that the TE of the *sound* synapse is floating. Next, during the learning phase, we simultaneously stimulate both the uncorrelated *sight of food* and *sound* neurons. The overlapping of the backward action pulses of the *salivation* neuron with the forward action pulses of the *sound* neuron thus create a voltage drop higher than the switching threshold

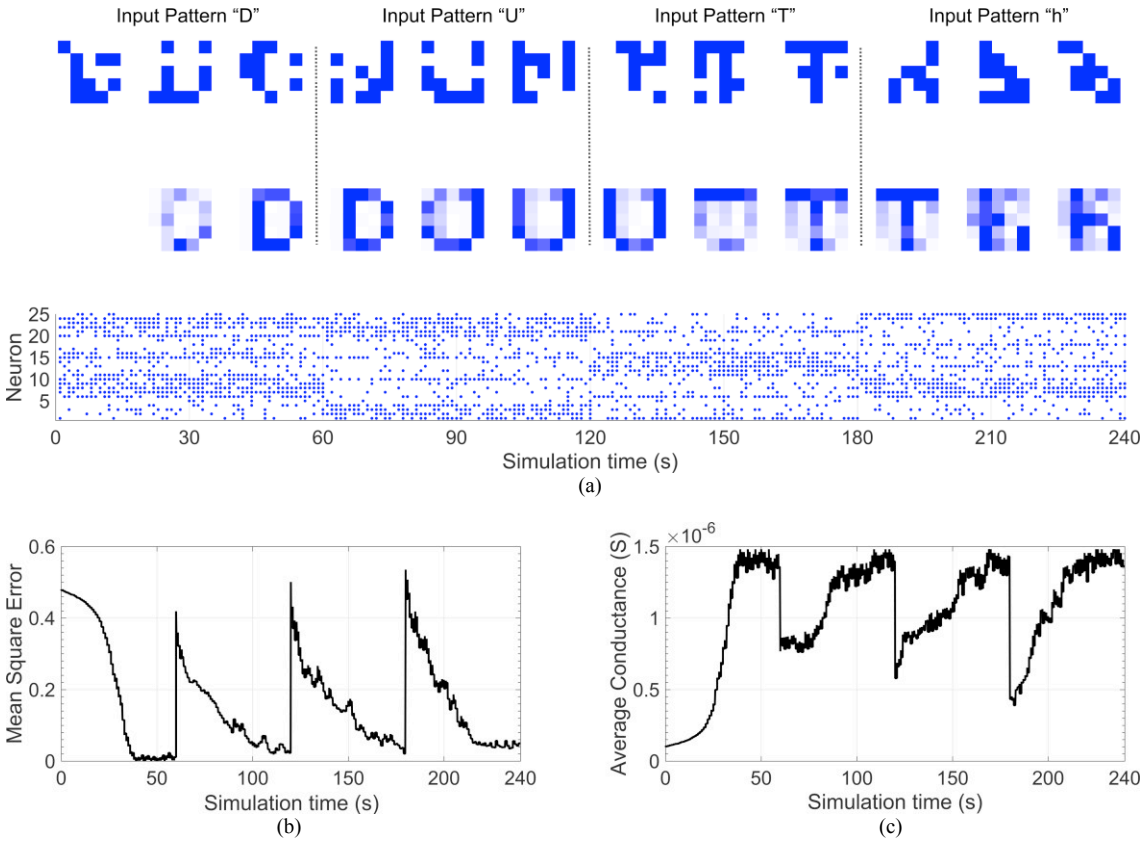


Fig. 13. HW simulation results. (a) Snapshots of the input images (first row) and the synaptic weights (second row) in the HW at the beginning, the middle, and the end of the 60s training period, respectively. White corresponds to low conductance and blue to high conductance. In the plot at the bottom, a dot corresponds to a firing neuron. (b) Time evolution of the mean square error of synaptic weights. (c) Time evolution of the average conductance of the target ON pixels of every pattern. The parameter values of the memristor emulator are set as in the previous example with α = -8Ω/(V×s), b = -128Ω/(V×s).

of the sound synapse (3V), hence causing its resistance to drop quickly. The final association of the two input neurons is demonstrated in the final probing phase, when any stimulus coming from either of the input neurons causes *salivation*. The different *salivation* neuron firing rate noticed in the last probing phase is simply attributed to the different conductance of the synapses (the sound synapse is being reinforced during this stage as its resistance keeps decreasing), causing an increasing V_{IN} value to the *salivation* neuron (see Fig. 9(c)) which consequently decreases the wait time of that neuron. Apparently, with a longer training phase the memductance of the sound neuron would eventually reach the G_{ON} value, so no change in the firing rate of the salivation neuron would be noted.

C. ANN Example for Unsupervised STDP Spatio-Temporal Pattern Learning

Following [37], which presents a correlation detector ANN applied in continuous STDP re-learning of spatio-temporal noisy patterns, we evaluated the developed ANN on a set of 5×5 pixel binary images used as input stimuli to 25 input neurons consecutively connected (row by row, starting from the top left pixel) to the 25 input pixels. Consequently there were 25 synapses connected to a single output neuron. Because of the simplified unsupervised STDP learning mechanism applied here [38], we show that in our HW ANN, the interaction between input and output neurons finally develops selectivity to particular features found in the noisy input image patterns. The ANN topology is very similar to that shown in Fig. 11 but now includes 25 input neurons (see Fig. 8). Unlike in the associative memory example, here it is also necessary to decrease the synaptic weights. Therefore, the V_{SIGN} output of the output neuron is now directly driven to the V_{SIGN} input of the KCL module to create negative and positive successive back propagating potentials. Moreover, according to [37], when the input neurons are not firing, the TE memristor terminals are considered grounded instead of floating. Here, the Valid_V_{TE} flag input of the memristor modules is thus constantly driven by value '1' to denote that it is not floating. Also, the V_{OUT} output of the input neurons is used as the selection bit of an intermediate multiplexer whose inputs are 2V and 0V, corresponding to an ON (dark) or OFF (white) pixel, respectively, and its output is driven to the V_{TE} memristor input. In short, when the output neuron fires, it applies a succession of a negative and a positive back propagating potential (see Fig. 9(c)). So, during the positive part, if there is no PRE pulse applied to the TE of a memristor (i.e., if it is grounded) there is a reverse voltage drop of amplitude $V_{POST} > V_T$, which decreases the synapse weight. Likewise, the synapse weight increase is performed in the same manner as in the associative memory example, when the forward voltage drop V_{PRE}-V_{POST} is higher than V_T.

In our experiment we consecutively presented temporally

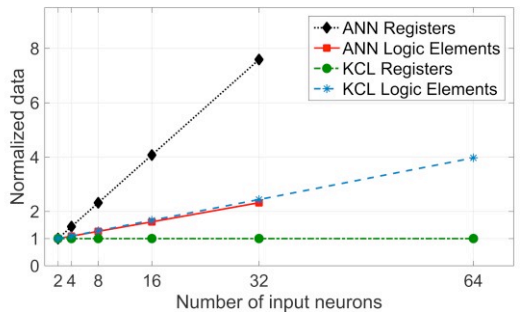


Fig. 14. FPGA HW resource scaling trends. KCL module only and ANN Total Registers/Logic Elements normalized to the 2-input case. The plot highlights the linear relationship of the KCL resources and the total ANN resources.

encoded signals derived from images of the letters 'D', 'U', 'T', and 'h', each for a training period of 60s, following the timescale of the original experiment in [37]. The input patterns are applied at time instances drawn from a Poisson distribution @1Hz rate. Furthermore, every pattern includes 20% Poissonian noise. Nevertheless, the correlated synapses progressively get potentiated, whereas the uncorrelated synapses are subject to depression. Thus, due to the simplified STDP learning, the ANN is finally able to extract the input pattern correctly. Moreover, when the input pattern changes, the ANN eventually re-learns the new information.

Fig. 13(a) presents the time-lapse sequence of the learning procedure for the four target patterns. The first row concerns the actual noisy input pattern and the second row corresponds to the actual weight values at that moment. For each letter we show the synapse weights at the beginning, middle, and end of the 60s training process. The plot at the bottom presents the state of the input neurons at every time step of the entire experiment, according to the applied input pattern. This plot highlights the effect of noise, also obvious in the first row of images. The results show that the ANN always learns the correlated features of the inputs even in the presence of important noise. Using such a simple STDP approach, the synaptic weights continuously store and update the learned information. Following [37], in Fig. 13(b) we show the time evolution of the mean square error (MSE) between each synapse weight and the target weight of the corresponding pixel of the input pattern. The highest MSE is seen at the beginning of the training process; however, it decreases to less than 1-2% by the time the training process is finished. Likewise, Fig. 13(c) shows the time evolution of the average conductance of the synapses corresponding only to the target ON pixels of every pattern. As can be seen, it approaches the maximum value during the training process. It decreases significantly when the input pattern changes due to changes in the ON pixels of interest. The local oscillations noticed during its evolution were simply attributed to the decrease in synapse weights due to noise; without noise, the average conductance monotonically increases during training.

D. FPGA HW Resource Scaling

This section aims to examine the HW resource scaling trends of the developed ANN FPGA implementations, as well as the required KCL module itself. This is because it would make little sense to consider memristive ANNs simulated on FPGAs if the KCL resources were predominant or if the total resources scaled unfavorably with regard to the number of ANN inputs.

TABLE IV
FPGA IMPLEMENTATION LOGIC DATA

***Cyclone II* (*EP2C70F672C6*)**		**Nº of Inputs**					
		2	**4**	**8**	**16**	**32**	**64**
KCL only	**Registers**	22	22	22	22	22	22
	Logic Elements	1338	1465	1721	2239	3260	5313
Full ANN	**Registers**	132	190	306	538	1002	N/A
	Logic Elements	2964	3227	3751	4789	6875	N/A

To this end, we collected the implementation data (total registers and logic elements) for the KCL only and for the complete ANN (see Fig. 10 and Fig. 11) with 2, 4, 8, 32, and 64 inputs for the Cyclone II target FPGA device, as shown in Table IV. However, unlike in Table I, here we used an optimized implementation using up to 14-bit registers for all the required variables, given that their value-range allowed this. Based on Table IV, Fig. 14 shows each data series normalized to its value for the 2-input ANN case, which is taken as a reference. Notably, as the number of inputs increases, the graphs clearly demonstrate a linear relation for all cases and, most importantly, with a small slope. Moreover, when comparing the KCL-only case with that of the full ANN, containing all the relevant presented modules, we noticed a ratio of less than 0.5 for the logic elements, and just a small constant number of registers required by the KCL.

The total HW resources for large ANNs are also target device-specific, whereas scaling will be much better in ASICs but here the focus is on FPGA devices, preferably for the reconfigurability they offer and the accessibility in academia, at considerably low costs. Our experiments showed that several n inputs-to-1 output (n×1) ANNs implemented on the same FPGA would require proportionally more HW. However, we found that ANNs with more than 32 inputs significantly increased the required FPGA resources. Fig. 14 does not include ANN data for 64 inputs as the synthesis would never be completed on the Cyclone II FPGA. This is attributed to the fact that such an FPGA has only 4 LUT inputs. Using a target device with more LUT inputs (e.g., one of the Cyclone V family), we implemented ANNs much larger than 64×1 without problem.

Thus, although we do not expect that the developed HW modules can be used to scale up the resulting ANNs to approach true brain density, owing to their simplicity, quite large ANNs can be implemented on proper target FPGAs using the modules developed here and thus achieve several ANN functionalities in various applications of interest.

V. CONCLUSIONS

The developed digital memristor HW simulator perfectly matches the mathematical model. Being the core of a true memristor digital emulator, it is interesting that it complies

with all the desired emulator requirements. Moreover, it can be readily used in complex circuit configurations and ANNs. It supports connection with other digital circuit modules due to the floating node flags and the use of additional HW, which computes the required node voltages (KCL module). Using an interface circuit will enable its connection with external circuitry, thus truly replacing a real memristor. The HW resource requirements are small and are linearly related to the number of ANN inputs. Passing from simulation in SW to emulation and execution in HW offers the major advantage of near-real time response. Following the presented approach, complex memristor circuit configurations and reasonably large ANNs can be easily implemented digitally for different applications. For this reason, all the corresponding VHDL descriptions are available to be used in research and/or in under/post-graduate student projects to enable an emulation-assisted HW exploration of memristive dynamics.

REFERENCES

[1] L. O. Chua, "Memristor - The Missing Circuit Element," *IEEE Trans. Circuit Theory*, vol. 18, no. 5, pp. 507-519, 1971

[2] L. O. Chua, "If it's pinched it's a memristor," *Semicond. Sci. Technol*., vol. 29, no. 10, pp. 104001, 2014

[3] S. D. Ha and S. Ramanathan, "Adaptive oxide electronics: a review," *J. Appl. Phys*., vol. 110, no. 7, pp. 071101, 2011

[4] T. W. Hickmott, "Low-Frequency Negative Resistance in Thin Anodic Oxide Films," *J. Appl. Phys*., vol. 33, no. 9, pp. 2669, 1962

[5] F. Argall, "Switching phenomena in titanium oxide thin films," *Solid State Electron*., vol. 11, no. 5, pp. 535-541, 1968

[6] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams, "The missing memristor found," *Nature*, vol. 453, pp. 80-83, May 2008

[7] "International Technology Roadmap for Semiconductors (ITRS)," 2013. [Online]. Available: http://www.itrs.net/. [Accessed June 2014]

[8] I. Vourkas and G. Ch. Sirakoulis, "Memristive Crossbar-Based Nonvolatile Memory," in *Memristor-Based Nanoelectronic Computing Circuits and Architectures*, 1st ed., Switzerland: *Springer Int. Publishing*, 2016, pp 101-147

[9] H.-S. P. Wong, *et al*., "Metal-Oxide RRAM," *IEEE Proc*., vol. 100, no. 6, pp. 1951-1970, 2012

[10] S. Paul and S. Bhunia, "A scalable memory-based reconfigurable computing framework for nanoscale crossbar," *IEEE Trans. Nanotechnol*., vol. 11, no. 3, pp. 451-462, 2012

[11] F. L. Traversa, Y. V. Pershin, and M. Di Ventra, "Memory Models of Adaptive Behavior," *IEEE Trans. Neural Netw. and Learning Syst*., vol. 24, no. 9, pp. 1437-1448, 2013

[12] A. Adamatzky and L. Chua (eds.), "Memristor Networks," *Springer Int. Publishing*, 2014, Switzerland, doi: 10.1007/978-3-319-02630-5

[13] J. J. Yang, D. B. Strukov, and D. R. Stewart, "Memristive devices for computing," *Nat. Nano*., vol. 8, pp. 13–24, 2013

[14] A. Siemon, *et al*., "Realization of Boolean Logic Functionality Using Redox-Based Memristive Devices," *Adv. Funct. Mater*., vol. 25, no. 40, pp. 6414–6423, 2015

[15] I. Vourkas and G. Ch. Sirakoulis, "Emerging Memristor-based Logic Circuit Design Approaches: A Review," *IEEE Circ. and Syst. Mag*, vol. 16, no. 3 (3rd quarter), pp. 15-30, 2016

[16] M. P. Sah, H. Kim, and L. O. Chua, "Brains are made of memristors," *IEEE Circ. and Syst. Mag*., vol. 14, no. 1, pp. 12-36, 2014

[17] D. Kuzum, S. Yu, and H-S P. Wong, "Synaptic electronics: materials, devices and applications," *Nanotechnology*, vol. 24, no. 38, pp. 382001, 2013

[18] D. Lin, S. Y. Hui, and L. O. Chua, "Gas Discharge Lamps Are Volatile Memristors," *IEEE Trans. Circ. Syst. I, Reg. Papers*, vol. 61, no. 7, pp. 2066-2073, 2014

[19] L. Chua, "Everything You Wish to Know About Memristors But Are Afraid to Ask," *Radioeng.*, vol. 24, no. 2, pp. 319-368, 2015

[20] S. H. Jo, T. Chang, I. Ebong, B. B. Bhadviya, P. Mazumder, and W. Lu, "Nanoscale memristor device as synapse in neuromorphic systems," *NANO Lett*., vol. 10, pp. 1297-1301, 2010

[21] S. Saighi, *et al.*, "Plasticity in memristive devices for spiking neural networks," *Front. Neurosci*., vol. 9 (51), March 2015

[22] M. Hu, H. Li, Y. Chen, Q. Wu, G. S. Rose, and R. W. Linderman, "Memristor Crossbar-Based Neuromorphic Computing System: A Case Study," *IEEE Trans. Neural Netw. and Learning Syst*., vol. 25, no. 10, pp. 1864 -1878, 2014

[23] S. Haykin, "Neural Networks: A Comprehensive Foundation," 2nd ed., *Prentice Hall*, NJ, USA, 1998

[24] C. Yakopcic, T. M. Taha, G. Subramanyam, and R. E. Pino, "Memristor SPICE Modeling," in *Advances in Neuromorphic Memristor Science and Applications*, 1st ed., *Springer Netherlands*, 2012, pp. 211-244

[25] A. Ascoli, F. Corinto, V. Senger, and R. Tetzlaff, "Memristor Model Comparison," *IEEE Circ. and Syst. Mag*., vol. 13, no. 2, pp. 89-105, 2013

[26] I. Vourkas, A. Batsos, and G. Ch. Sirakoulis, "SPICE modeling of nonlinear memristive behavior," *Int. J. Circ. Theor. Appl*., vol. 43, no. 5, pp. 553–565, 2015

[27] Y. V. Pershin and M. Di Ventra, "Practical Approach to Programmable Analog Circuits With Memristors," *IEEE Trans. Circ. Syst. I, Reg. Papers*, vol. 57, no. 8, pp. 1857-1864, 2010

[28] Bio Inspired Technologies, "Neuro-bit: the world's first commercially available memristor," [Online]. Available: http://www.bioinspired.net/ [Accessed 15 December 2015]

[29] D. Biolek, "Memristor Emulators," in *Memristor Networks*, A. Adamatzky, L. Chua (Eds.), *Springer Int. Publishing*, pp. 487-503, 2014

[30] C. Yang, H. Choi, S. Park, M. Pd Sah, H. Kim, and L. Chua, "A memristor emulator as a replacement of a real memristor," *Semicond. Sci. Technol.*, vol. 30 (015007), 2015

[31] Y. Pershin and M. Di Ventra, "Experimental demonstration of associative memory with memristive neural networks," *Neural Netw*., vol. 23, no. 7, pp. 881-886, 2010

[32] I. Vourkas, V. Ntinas, A. Abusleme, G. Ch. Sirakoulis, and A. Rubio, "A Digital Memristor Emulator for FPGA-Based Artificial Neural Networks," 2016 *IEEE Int. Verification and Security Workshop* (*IVSW*), Sant Feliu de Guixols, Catalunya, Spain, July 4 - 6, pp. 65-68

[33] J. Liu and D. Liang, "A Survey of FPGA-Based Hardware Implementation of ANNs," 2005 *Int. Conf. Neural Networks and Brain* (*ICNN&B*), vol. 2, Oct. 13-15 2005, Beijing, China, pp. 915 - 918

[34] A. R. Omondi and J. C. Rajapakse (Eds.), "FPGA Implementations of Neural Networks," *Springer*, Dordrecht, The Netherlands, 2006

[35] I. Vourkas and G. Ch. Sirakoulis, "Nano-Crossbar Memories Comprising Parallel/Serial Complementary Memristive Switches," *BioNanoSci*., vol. 4, no. 2, pp. 166-179, 2014

[36] T. Serrano-Gotarredona, T. Masquelier, T. Prodromakis, G. Indiveri, and B. Linares-Barranco, "STDP and STDP Variations with Memristors for Spiking Neuromorphic Learning Systems," *Front. Neurosci.*, vol. 7, no. 2, pp. 1-15, 18 Feb. 2013

[37] S. Wozniak, T. Tuma, A. Pantazi, and E. Eleftheriou, "Learning Spatio-Temporal Patterns in the Presence of Input Noise using Phase-Change Memristors," 2016 *IEEE Int. Symp. Circuits and Syst*. (*ISCAS*), Montreal, Canada, May 22-25, pp. 365 - 368

[38] D. Querlioz, O. Bichler, and C. Gamrat, "Simulation of a Memristor-Based Spiking Neural Network Immune to Device Variations," 2011 *Int. Joint Conference on Neural Networks* (*IJCNN*), San Jose, CA, USA, Jul. 31 - Aug. 5, pp. 1775-1781

[39] Q. Yu, H. Tang, K. Chen Tan, and H. Li, "Rapid Feedforward Computation by Temporal Encoding and Learning With Spiking Neurons," *IEEE Trans. Neural Netw. and Learning Syst*., vol. 24, no. 10, pp. 1539-1552, 2013

[40] J. Hu, H. Tang, K. C. Tan, and H. Li, "How the Brain Formulates Memory: A Spatio-Temporal Model," *IEEE Comp. Intelligence Mag*., vol. 11, no. 2, pp. 56-68, 2016

[41] I. Pavlov, "Conditioned reflexes: an investigation of the physiological activity of the cerebral cortex," *Oxford University Press*, London, 1927

Vasileios Ntinas received his M. Eng. Diploma and M.Sc. in electrical and computer engineering from Democritus University of Thrace (DUTh), Xanthi, Greece, in 2015 and 2017, respectively. He is currently a Ph.D. student in the Electrical and Computer Engineering Department at DUTh. His thesis focuses on the design and simulation of bio-inspired memristor-based computing circuits and systems.

Ioannis Vourkas (S'12-M'16) received his M. Eng. diploma and Ph.D. in electrical and computer engineering (ECE) from Democritus University of Thrace (DUTh), Xanthi, Greece, in 2008 and 2014, respectively. He is currently faculty member of the Department of Electronic Engineering, Universidad Técnica Federico Santa María, Valparaíso, Chile, where he is also PI in the project CONICYT FONDECYT Postdoctorado No. 3160042.

His current research emphasis is novel nano-electronic circuits and architectures comprising memristors. His research interests include unconventional computing, software and hardware aspects of parallel complex computational (bio-inspired) circuits and systems, and cellular automata. He is associate editor of Elsevier Microelectronics Journal.

He has also been a scholar at the Greek BODOSSAKI Foundation (2011 to 2014).

Angel Christian Abusleme Hoffman (M'01) was born in Santiago de Chile in 1975. He earned his diploma and M.SC. in electrical engineering from Pontificia Universidad Católica de Chile (PUC Chile) in 2000, and his Ph.D. in microelectronics from Stanford University, CA, USA, in 2011. He is currently an Associate Professor of Electrical Engineering at PUC Chile. His research has focused on instrumentation circuits for particle physics experiments.

Georgios Ch. Sirakoulis (M'95) received his M. Eng. diploma and Ph.D. in electrical and computer engineering (ECE) from Democritus University of Thrace (DUTh), Greece, in 1996 and 2001, respectively.

He is a Tenured Professor in the ECE Department at DUTh (2017-today). He has participated as a PI in more than 20 scientific programs and projects funded from the Greek government and industry and the EU Commission. His research interests include emergent electronic circuits and systems, memristors, green and unconventional computing, cellular automata, complex systems, and bio-inspired computation/bio-computation.

He serves as an associate editor of several peer reviewed international journals. He was General Co-Chair of the 2014 International Conference on Cellular Automata for Research and Industry (ACRI), and several other conferences, including ACRI 2012.

Antonio Rubio (SM'96) received his M.Sc. and Ph.D. from the Industrial Engineering Faculty, Polytechnic University of Catalonia (UPC), Barcelona, Spain.

He has been an Associate Professor in the Electronic Engineering Department, UPC, and a Professor in the Physics Department, Balearic Islands University. He is currently a Professor of electronic technology in the Telecommunication Engineering Faculty, UPC. His research interests include VLSI design and test, device and circuit modeling, high-speed circuit design, and new emerging nanodevices and nanoarchitectures.